AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listing, of claims in the application:

Listing of Claims:

1. (Currently amended) A method of creating a grammar for a natural language dialog system from a domain model, said method comprising the steps of:

a) creating a new grammar for the natural language dialog system from instances of general purpose grammar rules ~~as a new grammar~~, said general purpose grammar rules including a plurality of ~~selected~~ objects, each of said objects including one or more attributes;

b) creating ~~an~~ a query level umbrella rule for each broad category of queries in said general purpose grammar rules;

c) selectively including the objects as domain objects in said new grammar;

d) creating at least one domain object level umbrella ~~rules~~ rule for the ~~domain object~~ one or more attributes; and

e) selectively including the one or more attributes in said new grammar.

2. (Currently amended) A method as claim 1, wherein the step (a) of creating the new grammar comprises the steps of:

i) creating an initially empty new grammar;

ii) opening a template grammar, said template grammar including parameterized general purpose rules to be instantiated;

iii) creating instances of said general purpose grammar rules; and

iv) adding the instantiated general purpose rules to said initially empty new grammar.

3. (Currently amended) A method as in claim 1, wherein each said query level umbrella rule includes a domain-object-independent non-terminal in a left-hand side and a set of expansions of said non-terminal in a right-hand side.

4. (Original) A method as in claim 3, wherein each of said set of non-terminal expansions is a domain-specific instantiation of the broad category.

5. (Currently amended) A method as in claim 3, wherein the domain-object-independent non-terminal relates the query level umbrella rule to a broad category of rules.

Claims 6-9. (Cancelled)

10. (Currently amended) A method as in claim 1, wherein in step (e) the developer selects whether ~~an attribute~~ at least one of the attributes is included in the new grammar.

11. (Currently amended) A method as in claim 10, wherein if the developer selects an attribute for inclusion, said method further comprises the steps of:

f) including attribute name grammar rules naming ~~said~~ the attributes selected by the developer;

g) identifying whether ~~said named~~ each attribute selected by the developer is complex or simple.

12. (Currently amended) A method as in claim 11, wherein in step (g) if ~~an~~ the attribute selected by the developer is identified as complex, said method comprising the steps of:

i) creating grammar rules relating the object including said complex attribute to a subsidiary domain object; and

ii) adding said ~~created relating~~ grammar rules to said new grammar.

13. (Currently amended) A method as in claim 11, wherein in step (g) if ~~said~~ the attribute selected by the developer is identified as simple, said method further comprising the steps of:

i) creating grammar rules for ~~the~~ at least one atomic ~~values~~ value of the simple attribute;

ii) adding said ~~created atomic value~~ grammar rules for at least one atomic value to said new grammar;

iii) creating grammar rules requiring the name of the object including the simple attribute and the atomic value of the ~~domain~~ attribute; and

iv) adding said grammar rules created in step (iii) to said new grammar.

14. (Currently amended) A method as in claim 1, said method further comprising the step of:

f) detecting and repairing inconsistencies in the ~~newly created~~ new grammar.

15. (Currently amended) A method as in claim 14, wherein the step (f) of detecting and repairing inconsistencies comprises the steps of:

i) running a grammar checker on the new grammar, said grammar checker identifying unreachable non-terminals;

ii) repairing ~~said newly created~~ the new grammar to eliminate identified unreachable non-terminals thereby creating a repaired new grammar;

iii) running said grammar checker on said repaired new grammar, said grammar checker identifying non-terminating expansions; and

iv) repairing said repaired new grammar to eliminate identified non-terminating expansions.

16. (Original) A method as in claim 15 wherein the step (ii) of eliminating unreachable non-terminals comprises the steps of:

A) prompting the developer to delete unreachable non-terminals; and

B) prompting the developer to add new rules to make all remaining non-terminals reachable.

17. (Original) A method as in claim 15, wherein the step (ii) of eliminating non-terminating expansions comprises the steps of:

A) prompting the developer to delete identified non-terminating expansions; and

B) prompting the developer to add rules terminating all remaining said non-terminating expansions.

18. (Currently amended) A system for creating a natural language dialog grammar from a domain model, said system comprising:

means for forming a new grammar for a natural language dialog system including instances of general purpose grammar rules, said instances of general purpose rules being a plurality of objects, each of said objects including one or more attributes;

means for creating query level umbrella rules for query categories and attributes;

means for selecting objects included as domain objects in said new grammar;

means for selecting attributes included in said new grammar; and

means for detecting and repairing inconsistencies in said new grammar.

19. (Currently amended) A system as in claim 18, wherein the means for forming the new grammar comprises:

means for creating an initially empty new grammar;

means for opening a template grammar including parameterized general purpose rules;

means for creating instances of general purpose rules; and

means for adding the general purpose rules to said initially empty new grammar.

20. (Currently amended) A system as in claim 19, wherein said means for creating query level umbrella rules creates a rule with a left-hand side being a domain-object-independent non-terminal and a right-hand side including a set of expansions of said non-terminal, each of said set of non-terminal expansions being a domain-specific instantiation of the broad category, said non-terminal relating the rule to a broad category of rules.

21. (Currently amended) A system as in claim 18, wherein the means for selectively including objects comprises:

means for selecting selected objects;

means for presenting said selected objects to a developer, said developer deciding whether said presented selected objects are included in said new grammar.

22. (Currently amended) A system as in claim 21, further comprising:

means for creating ~~an~~ a domain object level umbrella rule for a broad category of object phrases;

means for creating object name grammar rules for the domain object level umbrella rule ~~rules~~;

means for receiving a name selection from a developer; and

means for entering names in said new grammar responsive to selection by said developer.

23. (Original) A system as in claim 22, wherein the means for creating object name rules further includes means for creating an object name possessive rule, said object name possessive rule specifying the possessive form of the name of said object.

24. (Original) A system as in claim 23, said system further comprising:

means for automatically specializing object rules; and

means for specializing query rules responsive only to the name of the domain object.

25. (Original) A system as in claim 18, wherein the means for selecting attributes included in the new grammar comprises:

means for presenting said attributes to a developer, said developer deciding whether each of said presented attributes is included.

26. (Currently amended) A system as in claim 25, said system further comprising:

means for generating attribute name grammar rules for ~~selected~~ said attributes selected by the developer;

complex attribute grammar rule creation means for identifying attributes, relating the ~~object~~ identified attributes to a subsidiary domain object as complex attributes, creating complex attribute grammar rules ~~being created~~ for said complex ~~attributes and added~~ attributes, and adding the complex attribute grammar rules to said new grammar;

atomic value grammar rule creation means for identifying simple attributes, creating atomic value grammar rules for atomic values of identified simple attributes and, adding said atomic value grammar rules to said new grammar; and

simple attribute grammar rule creation means for creating simple attribute grammar rules requiring the name of simple objects and a corresponding atomic value and adding said simple attribute grammar rules to said new grammar.

27. (Original) A system as in claim 18, wherein the means for detecting and repairing inconsistencies comprises:

means for identifying unreachable non-terminals;

means for eliminating unreachable non-terminals from said new grammar;

means for identifying non-terminating expansions; and

means for eliminating non-terminating expansions from said new grammar.

28. (Original) A system as in claim 27, wherein the means for eliminating unreachable non-terminals comprises:

means for allowing a developer to delete identified unreachable non-terminals; and

means for allowing the developer to add new rules making identified unreachable non-terminals reachable.

29. (Original) A system as in claim 28, wherein the means for eliminating non-terminating expansions comprises:

means for allowing the developer to delete identified non-terminating expansions; and

means for allowing the developer to add rules terminating said identified non-terminating expansions.

30. (Currently amended) A computer program product for creating a natural language dialog grammar from a domain model, said computer program product comprising a computer usable medium having computer readable program code thereon, said computer readable program code comprising:

computer readable program code means for forming a new grammar <u>for a natural language dialog system</u> including instances of general purpose grammar rules, said instances of general purpose rules being a plurality of objects, each of said objects including one or more attributes;

computer readable program code means for creating query level umbrella rules~~, umbrella rules being created~~ for query categories and for attributes; and

computer readable program code means for selecting objects and attributes included as domain objects and domain object attributes in said new grammar.

31. (Currently amended) A computer readable program code means for creating a natural language grammar from domain models as in claim 30, wherein the computer readable program code means for forming the new grammar comprises:

computer readable program code means for creating an initially empty new grammar;

computer readable program code means for opening a template grammar including parameterized general purpose rules to be instantiated;

computer readable program code means for creating instances of general purpose rules; and

computer readable program code means for adding the instantiated general purpose rules to said initially empty new grammar.

32. (Currently amended) A computer readable program code means for creating a natural language grammar from domain models as in claim 31, wherein said computer readable program code means for creating query level umbrella rules creates rules with a left-hand side being a domain-object-independent non-terminal and a right-hand side including a set of expansions of said non-terminal, each of said set of non-terminal expansions being a domain-specific instantiation of the broad category, said non-terminal relating the rule to a broad category of rules.

33. (Currently Amended) A computer readable program code means for creating a natural language grammar from domain models as in claim 32, wherein the computer readable program code means for selecting objects and attributes comprises:

computer readable program code means for individually selecting selected objects; and

computer readable program code means for presenting said selected objects to a developer, said developer deciding whether said presented selected objects are included in said new grammar.

34. (Currently amended) A computer readable program code means for creating a natural language grammar from domain models as in claim 33, wherein the computer readable program code means for selecting objects and attributes further comprises:

computer readable program code means for presenting ~~selected~~ said attributes to a developer, said developer deciding whether each of said presented attributes is included in said new grammar.

35. (Currently amended) A computer readable program code means for creating a natural language grammar from domain models as in claim 34~~, wherein the computer readable program code means for creating umbrella rules further creates umbrella rules for a broad category of object phrases, said computer readable program code means~~ further comprising:

computer readable program code means for creating object name grammar rules for at least one domain object level umbrella rule ~~rules~~;

computer readable program code means for receiving a name selection from a developer, said name selection selectively including said object name created by said computer responsive to creating said object name grammar rules; and

computer readable program code means for entering names in said new grammar responsive to selection by said developer.

36. (Currently amended) A computer readable program code means for creating a natural language grammar from domain models as in claim 35, wherein the computer readable program code means for creating object name grammar rules further includes computer readable program code means for creating an object name possessive rule, said object name possessive rule specifying the possessive form of the name of said object.

37. (Original) A computer readable program code means for creating a natural language grammar from domain models as in claim 36, said system further comprising:

computer readable program code means for automatically specializing object rules; and

computer readable program code means for specializing query rules responsive only to the name of the domain object.

38. (Currently amended) A computer readable program code means for creating a natural language grammar from domain models as in claim 35, further comprising:

computer readable program code means for generating attribute name grammar rules for selected said attributes;

computer readable program code means for identifying attributes relating to a subsidiary domain object as complex attributes, creating complex attribute grammar rules ~~being created~~ for said complex ~~attributes and added~~ attributes, and adding the complex attribute grammar rules to said new grammar;

computer readable program code means for identifying simple attributes, creating atomic value grammar rules for atomic values of simple attributes, and adding said atomic value grammar rules to said new grammar; and

computer readable program code means for creating simple attribute grammar rules requiring the name of simple objects and a corresponding atomic value and, adding said simple attribute grammar rules to said new grammar.

39. (Currently amended) A computer readable program code means for creating a natural language grammar from domain models as in claim 30, said system further comprising:

computer readable program code means for detecting and repairing inconsistencies in ~~newly created~~ the new grammar.

40. (Original) A computer readable program code means for creating a natural language grammar from domain models as in claim 39, wherein the computer readable program code means for detecting and repairing inconsistencies comprises:

computer readable program code means for identifying unreachable non-terminals;

computer readable program code means for eliminating unreachable non-terminals;

computer readable program code means for identifying non-terminating expansions; and

computer readable program code means for eliminating non-terminating expansions.

41. (Original) A computer readable program code means for creating a natural language grammar from domain models as in claim 40, wherein the computer readable program code means for eliminating unreachable non-terminals comprises:

computer readable program code means for selectively deleting identified unreachable non-terminals; and

computer readable program code means for selectively adding new rules making identified unreachable non-terminals reachable.

42. (Original) A computer readable program code means for creating a natural language grammar from domain models as in claim 41, wherein the computer readable program code means for eliminating non-terminating expansions comprises:

computer readable program code means for selectively deleting identified non-terminating expansions; and

computer readable program code means for selectively adding rules terminating identified non-terminating expansions.

43. (New) A method of creating a grammar for a natural language dialog system from a domain model, said method comprising the steps of:

a) creating a new grammar for the natural language dialog system from instances of general purpose grammar rules, said general purpose grammar rules including a plurality of objects, each of said objects including one or more attributes;

b) creating a query level umbrella rule for each broad category of queries in said general purpose grammar rules;

c) selectively including the objects as domain objects in said new grammar comprising the steps of:

i) selecting a selected object from said plurality of objects; and

ii) presenting said selected object to a developer, said developer deciding whether said selected object is included in said new grammar;

d) creating at least one domain object level umbrella rule for the attributes; and

e) selectively including the attributes in said new grammar.

44. (New) A method as in claim 43, wherein if said developer decides that said selected object should be included in said grammar, said method further comprises the steps of:

c1) creating a domain object level umbrella rule for each broad category of object phrases;

c2) creating an object name grammar rule for each created domain object level umbrella rule;

c3) allowing the developer to select names for the selected object; and

c4) adding entries in said grammar for each selected name.

45. (New) A method as in claim 44, wherein the object name grammar rules created in step (c2) further include an object name possessive rule, said object name possessive rule specifying the possessive form of the name of said object.

46. (New) A method as in claim 45, said method further comprising the steps of:

c5) specializing object rules not requiring developer input; and

c6) specializing query rules requiring only the name of the domain object.

47. (New) A method as claim 43, wherein the step (a) of creating the new grammar comprises the steps of:

i) creating an initially empty new grammar;

ii) opening a template grammar, said template grammar including parameterized general purpose rules to be instantiated;

iii) creating instances of said general purpose grammar rules; and

iv) adding the instantiated general purpose rules to said initially empty new grammar.

48. (New) A method as in claim 43, wherein each said query level umbrella rule includes a domain-object-independent non-terminal in a left-hand side and a set of expansions of said non-terminal in a right-hand side.

49. (New) A method as in claim 48, wherein each of said set of non-terminal expansions is a domain-specific instantiation of the broad category.

50. (New) A method as in claim 48, wherein the domain-object-independent non-terminal relates the query level umbrella rule to a broad category of rules.

51. (New) A method as in claim 43, wherein in step (e) the developer selects whether at least one attribute is included in the new grammar.

52. (New) A method as in claim 51, wherein if the developer selects an attribute for inclusion, said method further comprises the steps of:

f) including attribute name grammar rules naming the attributes selected by the developer;

g) identifying whether each attribute selected by the developer is complex or simple.

53. (New) A method as in claim 52, wherein in step (g) if the attribute_selected by the developer is identified as complex, said method comprising the steps of:

i) creating grammar rules relating the object including said complex attribute to a subsidiary domain object; and

ii) adding said grammar rules to said new grammar.

54. (New) A method as in claim 52, wherein in step (g) if the attribute selected by the developer is identified as simple, said method further comprising the steps of:

i) creating grammar rules for at least one atomic value of the simple attribute;

ii) adding said grammar rules for at least one atomic value to said new grammar;

iii) creating grammar rules requiring the name of the object including the simple attribute and the atomic value of the attribute; and

iv) adding said grammar rules created in step (iii) to said new grammar.

55. (New) A method as in claim 43, said method further comprising the step of:

f) detecting and repairing inconsistencies in the new grammar.

56. (New) A method as in claim 55, wherein the step (f) of detecting and repairing inconsistencies comprises the steps of:

i) running a grammar checker on the new grammar, said grammar checker identifying unreachable non-terminals;

ii) repairing the new grammar to eliminate identified unreachable non-terminals thereby creating a repaired new grammar;

iii) running said grammar checker on said repaired new grammar, said grammar checker identifying non-terminating expansions; and

iv) repairing said repaired new grammar to eliminate identified non-terminating expansions.

57. (New) A method as in claim 56 wherein the step (ii) of eliminating unreachable non-terminals comprises the steps of:

A) prompting the developer to delete unreachable non-terminals; and

B) prompting the developer to add new rules to make all remaining non-terminals reachable.

58. (New) A method as in claim 56, wherein the step (ii) of eliminating non-terminating expansions comprises the steps of:

A) prompting the developer to delete identified non-terminating expansions;

and

B) prompting the developer to add rules terminating all remaining said non-terminating expansions.